



## Freeform Search

---

<b>Database:</b>	US Pre-Grant Publication Full-Text Database
	US Patents Full-Text Database
	US OCR Full-Text Database
	EPO Abstracts Database
	JPO Abstracts Database
	Derwent World Patents Index
	IBM Technical Disclosure Bulletins

<b>Term:</b>	<input type="text"/>	
		

<b>Display:</b>	<input type="text" value="10"/>	<b>Documents in Display Format:</b>	<input type="text" value="-"/>	<b>Starting with Number</b>	<input type="text" value="1"/>
-----------------	---------------------------------	-------------------------------------	--------------------------------	-----------------------------	--------------------------------

**Generate:** ☐ Hit List ☒ Hit Count ☐ Side by Side ☐ Image

---

Search

Clear

Interrupt

---

### Search History

---

**DATE:** Tuesday, September 14, 2004    [Printable Copy](#)    [Create Case](#)

**Set Name Query**

side by side

**Hit Count Set Name**

result set

*DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR*

<u>L24</u>	14 and 116	62	<u>L24</u>
<u>L23</u>	14 and 115	57	<u>L23</u>
<u>L22</u>	707/104.1	4578	<u>L22</u>
<u>L21</u>	707/100	5107	<u>L21</u>
<u>L20</u>	707/200	3482	<u>L20</u>
<u>L19</u>	707/202	1822	<u>L19</u>
<u>L18</u>	707.clas.	22378	<u>L18</u>
<u>L17</u>	705.clas.	28949	<u>L17</u>
<u>L16</u>	705/1	5043	<u>L16</u>
<u>L15</u>	705/35	2096	<u>L15</u>
<u>L14</u>	L13 and investment near advisor	10	<u>L14</u>
<u>L13</u>	portfolio near manag\$ and traders	159	<u>L13</u>
<u>L12</u>	111 and server	39	<u>L12</u>
<u>L11</u>	L10 and manag\$	39	<u>L11</u>
<u>L10</u>	L9 and business near3 logic	39	<u>L10</u>
<u>L9</u>	L8 and session near manag\$	71	<u>L9</u>

<u>L8</u>	L4 and (data with base or database)	706	<u>L8</u>
<u>L7</u>	L5 and shar\$ near (database or data with base)	22	<u>L7</u>
<u>L6</u>	L5 and (shar\$ or coherency) near (database or data with base)	22	<u>L6</u>
<u>L5</u>	L4	766	<u>L5</u>
<u>L4</u>	L3 and (user near profile or user near history)	766	<u>L4</u>
<u>L3</u>	(business or financial) near transaction	11741	<u>L3</u>
<u>L2</u>	L1 and (business or financial) near transaction	0	<u>L2</u>
<u>L1</u>	database near coherency	27	<u>L1</u>

END OF SEARCH HISTORY

[First Hit](#) [Fwd Refs](#)[Previous Doc](#)[Next Doc](#)[Go to Doc#](#)

Generate Collection

Print

L8: Entry 3 of 4

File: USPT

Sep 2, 2003

DOCUMENT-IDENTIFIER: US 6615258 B1

TITLE: Integrated customer interface for web based data management

Detailed Description Text (4):

FIG. 1 is a diagrammatic illustration of the software architecture component in which the present invention functions. A first or client tier 10 of software services are resident on a customer workstation 10 and provides customer access to the enterprise system, having one or more downloadable application objects directed to front-end business logic, one or more backplane service objects for managing sessions, one or more presentation services objects for the presentation of customer options and customer requested data in a browser recognizable format and a customer supplied browser for presentation of customer options and data to the customer and for communications over the public Internet. Additional applications are directed to front-end services such as the presentation of data in the form of tables and charts, and data processing functions such as sorting and summarizing in a manner such that multiple programs are combined in a unified application suite.

Detailed Description Text (9):

The Customer Browser 20, is a Web browser which is Java-enabled and includes client applications responsible for presentation and front-end services. Its functions include providing a user interface to various data management services and supporting communications with the enterprise's Web server cluster 24. The client tier software is responsible for presentation services to the customer and generally includes a Web browser 14 and additional object-oriented programs residing in the client workstation platform 20. The client software is generally organized into a component architecture with each component generally comprising a specific application, providing an area of functionality. The applications generally are integrated using a "backplane" services layer 12 which provides a set of services to the application objects that provide the front-end business logic. The backplane services layer 12 also manages the launching of the application objects. The common set of objects provide a set of services to each of the applications. The set of services include: 1) session management; 2) application launch; 3) inter-application communications; 4) window navigation among applications; 5) log management; and 6) version management.

Detailed Description Text (12):

As will be hereinafter described in greater detail a customer session is designated by a logon, successful authentication, followed by use of server resources, and logoff. However, the world-wide Web communications protocol uses HTTP, a stateless protocol, each HTTP request and reply is a separate TCP/IP connection, completely independent of all previous or future connections between the same server and client. The system of the present invention is implemented with a secure version of HTTP such as S-HTTP or HTTPS, and preferably utilizes the SSL implementation of HTTPS. The preferred embodiment uses SSL which provides a cipher spec message which provides server authentication during a session. The preferred embodiment further associates a given HTTPS request with a logical session which is initiated and tracked by a "cookie jar server" 28 to generate a "cookie" which is a unique server-generated key that is sent to the client along with each reply to a HTTPS request. The client holds the cookie and returns it to the server as part of each subsequent HTTPS request. As desired, either the Web servers 24, the cookie jar

server 28 or the Dispatch Server 26, may maintain the "cookie jar" to map these keys to the associated session. A separate cookie jar server 28, as illustrated in FIG. 2 has been found desirable to minimize the load on the dispatch server 26. This form of session management also functions as an authentication of each HTTPS request, adding an additional level of security to the overall process.

Detailed Description Text (27):

When the backplane is implemented as an applet, it overrides standard Applet methods `init( )`, `start( )`, `stop( )` and `run( )`. In the `init( )` method, the backplane applet obtains a COUser user context object. The COUser object holds information such as user profile, applications and their entitlements. The user's configuration and application entitlements provided in the COUser context are used to construct the application toolbar and Inbox applications. When an application toolbar icon is clicked, a particular COApp is launched by `launchapp( )` method. The launched application then may use the backplane for inter-application communications, including retrieving Inbox data.

Detailed Description Text (52):

The present invention also includes a user unit for representing a user of a current session. The user unit is generally implemented as a COUser class extending `java.lang.Object`. The COUser class object holds information including a user profile, applications and their entitlements. In order to minimize network traffic, the amount of data carried by the COUser is minimal initially, and get populated as requests are processed. The requests are generally processed by retrieving information from the OE service. The profile information is then stored and populated in the COUser object should such information be requested again.

Detailed Description Text (53):

A COUser object is created when the user logs in, and holds the username and password of the user as an object in the COClientSession object. The session object is contained within the backplane, which manages the session throughout its lifetime. The code below illustrates how this occurs: `// Within the backplane  
COClientSession session=new COClientSession( ); try { Session. logon ("username",  
"password"); } catch (COClientLogonException e) { . . . }; // Should the User object  
be required COUser user=session.getuser( );`

Detailed Description Text (63):

When a customer launches the OE application from the home page, the main window as illustrated in FIG. 14, is presented. From this main window 1500, a customer may select to order and fulfill application services, request user identifiers (ids), and create user security profiles for the data management suite of applications. The main window 1500 includes a menu bar 1506 with options to perform various OE tasks. The main window also includes a toolbar 1504, common to all data management applications. The toolbar 1504 has buttons that also perform the various OE functions. Typically, the user list is presented, i.e., displayed as a tree 1502, within the main window 1500.

Detailed Description Text (72):

In the preferred embodiment, as shown in FIG. 7, the OE server 39 provides a number of processes for performing a number of specific functions. For example, a fulfillment process monitors new customers being added to the system and notifies a fulfillment house 298 accordingly (FIG. 9). The fulfillment house then may send appropriate subscription packages according to the information received from the fulfillment process to the new customer. Another process, a reconciliation process, may handle synchronization of data with a mainframe system database and also with databases associated with the individual fulfilling systems. Yet another process, a billing process, may handle directing billing information to different billing streams 157 (FIG. 7).

Detailed Description Text (74):

Referring to FIG. 7, a process running in a OE client application process 154 sends transaction request messages via the infrastructure, comprising, e.g., the Web server cluster 24 and a dispatch server 26 (FIG. 2), to the OE server 39. The OE server 39 responds to requests by searching the security profile for the information requested, formulating appropriate transaction response messages and transmitting them back to the requesting process. As an example, during the login procedure, the client login process formulates a transaction message including a user name/password and a validation request for a given customer. The OE server 39 looks for the matching name/password pair in the security profile for the customer, and if the name/password pair is found, the server 39 formulates a valid user message response for the login process running in the client platform, including in the message the enterprise id, time zone, and user id information and transmits the response via TCP/IP back to the login process. When the OE server 39 detects that the password has expired, the server 39 notifies the customer, via the client application 154 to change the password. The changed password is sent to the OE server 39 formatted in a message interface, "change password request," for example. The server 39 upon receiving the message updates the password for the given user in its user profile stored in OE database 160, and responds with appropriate return codes to the OE client 154. The login process, upon receiving the response may then continue with its normal course of processing.

Detailed Description Text (77):

In providing the authentication, entitlement, and hierarchy information, including those described above, the OE server database 160 stores user profiles locally. Additional data needed are typically accessed from the enterprise host systems 159. The OE server 39 may be implemented as a concurrent server allowing simultaneous multiple client connections. The server 39 listens on a known port, and when a client connects, the server 39 forks a new process to handle the client connection. The server 39 may also implement a thread to handle each client connection.

Detailed Description Text (84):

FIG. 9 is an output process flow diagram, illustrating outputs and responses from the OE server 39 to the requesting systems and processes. An example of an output is an authentication response to the client side of the individual applications, e.g., reporting system 400, etc., as well as the backplane. In addition, a list of accessible applications for a given customer, is output to the backplane platform via platform Web servers 24. The OE also outputs various updated data to database systems associated with specific individual applications in the suite of data management applications. In addition, the individual fulfilling systems receive messages from the OE regarding modifications effected by a customer interaction. For example, the customer hierarchy data is sent in real time by the OE for up-to-date report information.

Detailed Description Text (157):

As in any of the above-described suite of data management applications, the Trouble Ticketing application utilizes the Common Objects application framework (COF) to inter-operate with the backplane and integrate with the other elements of the architecture provided in the system of the present invention. The Common Objects framework is utilized to leverage existing infrastructure services such as login and authentication, transaction management, and security. Particularly, the Trouble Ticketing application extends the COAppImpl class in order to inter-operate with the backplane and other applications (as required), and, includes one or more screens derived from the COAppFrame class. Most of the high level classes dealing with the initiation of transactions are utilized by Trouble Ticketing. The COClientSession class is available to the Trouble Ticketing application upon successful login to the system of the present invention and is utilized for session management (e.g., connect, disconnect, and logoff). The family of COTransaction classes is used to send and receive messages to the back-end Trouble Ticketing service. These classes include CONonblockTransaction, COSynchTransaction, and COAsynchTransaction and, a COBulkTransaction may also be used if necessary.

Additionally, the Trouble Ticketing utilizes all of the COCommunications classes with the exception of the COBulkTransaction. However, as development and testing continues, the COBulkTransactions class may be utilized.

Detailed Description Text (158):

FIG. 16(a) illustrates the high-level design of the Trouble Ticketing application 2200 including the client application 2250 and server 2300 components. As shown, Trouble Ticketing requires integration with a number of external systems and utilizes the Common Objects Framework for inter-application communications. Interfacing with the Trouble Ticketing application server 36 via the common objects framework are the OE server, e.g., for user profile information, as well as other Trouble Ticketing specific data, and, the CSM legacy host that provides the ability to query, status, and take action on service inquiries. Communication between the Trouble Ticketing application server 36 and CSM 40(a) is via Registry middleware, such as described in commonly owned, co-pending U.S. patent application Ser. No. 08/560,550 incorporated by reference herein. FIG. 3 shows COF-based inter-application communication between Trouble Ticketing and OE. It should be understood that if an external system does not use the COF, Trouble Ticketing may utilize that system's API set and communication mechanism for inter-application communication. The above-referenced Registry system has a number of options for inter-application communication, including both Java and CORBA interfaces.

Detailed Description Text (160):

The Trouble Ticketing application Server 2300 interfaces with the Legacy Backend 40 (a), CSM through a Requester object 2310 and Receiver object 2350 as shown in FIG. 16(b). Particularly, the SvcInqCSMRequester object 2310 is the class that represents the requester which takes the request data that comes from the Front-End/Client application through the Transaction Manager 2320, builds the CSM request transactions by interacting with the Translator classes 2380 and ships off the requests to CSM. The request data that comes from the Front End/Client is an array of strings that are required from the customer for the request to be made. Minimal information is passed from the client to reduce the communication overhead from the client to the Trouble Ticketing application server. All other information is packaged in the Requester. Particularly, the Requester object 2310 uses the SvcInqRegistryHeader and SvcInqSIHeader classes in the Translator 2380 to build the "Registry Header" and "Trouble Ticketing Header" strings that are required for the CSM request transactions. It also talks to the SvcInqActivity or the SvcInqRemarks classes to build the data portion of the CSM requests. Once the CSM Transaction String is formatted the actual request to CSM is made. Sending the transaction to CSM's Standard Interface (SI) via Registry classes does this.

Detailed Description Text (161):

The receiver object is an instance of the SIRegistryHandler class whose responsibility is to obtain the responses from CSM, parse the response, strip off the headers and build objects from the response data, by interacting with the Translator classes 2380. Particularly, it uses the SvcInqRemark, the SvcInqActivity, the SvcInqTroubleTicket or the SvcInqRegistryEntry class in the Translator to build the remark, activity, detail or list of Ticket object from the response string that is received from CSM. The built object is then sent back to the Transaction Manager 2380 who passes it back to the Front-End/Client.

Detailed Description Text (169):

As an example, a "List Tickets by Status Request" transaction will provide all the tickets for a given organization (ORG) code with the requested status and created after a specified date. The ORG code to be passed in this transaction is one of the selection criteria representing the originating organization or the organization where the ticket was created. The customer may choose from a list of ORGs that the customer has authority over and a primary ORG is obtained from every customer and is stored locally in the user profile. The resulting information from all of the tickets will be cached for future processing. Generally, only one type of status

may be specified in a single request: Open, Closed, Referred or Canceled status. If a customer has authority over more than one organization that customer is able to view tickets for any organization he/she has authority over. If a customer has access to a primary organization, then he/she has implied access to all the subordinate organizations meaning that the request will apply to the subordinate organizations as well. Furthermore, this transaction may only display some of the details/fields of the tickets which means that the data cached from this request may only be used to process the Queries on tickets. It cannot be used to view all the details of the tickets for which further CSM transactions will have to be made as will be herein described.

Detailed Description Text (178):

Once the ticket is opened, it has to be referred out to a "Customer Facing Organization" to initiate the problem resolution process. To do this, the CSM system refers the ticket out to an organization obtained from the user up front and stored in the User Profile. This is done using an "Enter Activity Request Transaction" which allows the customer to enter different activities like 'Refer Out', 'Close', 'Refer Back' and 'Open' on a ticket by passing the appropriate activity code.

Detailed Description Text (265):

When a user is authenticated at login via the system administrative server, the client session object is given a "cookie", a unique server-generated key which identifies a session. The session key is typically encapsulated in a class COWebCookie, "public CoWebCookie (int value).", where value represents a given cookie's value. The client session object holds this key and returns it to the server as part of the subsequent HTTP request. The Web server maintains a "cookie jar" which is resident on the dispatch server and which maps these keys to the associated session. This form of session management also functions as an additional authentication of each HTTPS request, adding security to the overall process. In the preferred embodiment, a single cookie typically suffices for the entire session. Alternatively, a new cookie may be generated on each transaction for added security. Moreover, the cookie jar may be shared between the multiple physical servers in case of a failure of one server. This mechanism prevents sessions being dropped on a server failure.

Other Reference Publication (14):

Lee et al., "Supporting Multi-User, Multi-Applet Workspaces in CBE", Computer Supported Cooperative Work 1996, Cambridge, MA.

CLAIMS:

2. The integrated data management system as claimed in claim 1, wherein the one or more secure servers support a secure sockets layer communications protocol for encrypted communication between the user interface, including the client applications, and the secure servers, the secure servers also providing session management including client identification, validation and session management to link the session with the customer.

26. The integrated data management system as claimed in claim 2, wherein said session management provided by the one or more secure servers includes Web cookie generation at each instance of client identification to link a session with the customer through a plurality of discrete client communications in the session to verify the customer to the dispatch server at each transmission in the session.

28. The integrated data management system as claimed in claim 27, wherein the secure socket layer encrypts client identification, authentication and the session management cookie during each transmission.

29. The integrated data management system as claimed in claim 28, wherein the

session cookies provide simultaneous session management for a plurality of system resource platforms.

60. The method as claimed in claim 59, further including: encrypting client identification, authentication and the session management cookie during each transmission.

61. The method as claimed in claim 60, wherein the method further includes: managing sessions simultaneously with the session cookies for a plurality of system resource platforms.

[Previous Doc](#)

[Next Doc](#)

[Go to Doc#](#)





## Freeform Search

---

<b>Database:</b>	US Pre-Grant Publication Full-Text Database
	US Patents Full-Text Database
	US OCR Full-Text Database
	EPO Abstracts Database
	JPO Abstracts Database
	Derwent World Patents Index
	IBM Technical Disclosure Bulletins

<b>Term:</b>	<input type="text"/>	
		

<b>Display:</b>	<input type="text" value="10"/>	<b>Documents in Display Format:</b>	<input type="text" value="-"/>	<b>Starting with Number</b>	<input type="text" value="1"/>
-----------------	---------------------------------	-------------------------------------	--------------------------------	-----------------------------	--------------------------------

**Generate:** ☐ Hit List ☒ Hit Count ☐ Side by Side ☐ Image

---

Search

Clear

Interrupt

---

### Search History

---

**DATE:** Tuesday, September 14, 2004    [Printable Copy](#)    [Create Case](#)

<u>Set</u> <u>Name</u>	<u>Query</u>	<u>Hit</u> <u>Count</u>	<u>Set</u> <u>Name</u> result set
side by side			
<i>DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR</i>			
<u>L8</u>	L7 and session near manag\$	4	<u>L8</u>
<u>L7</u>	L6 and transaction near manag\$	77	<u>L7</u>
<u>L6</u>	L5 and (user with profiles or user near profiles or history with profiles or history with profiles)	232	<u>L6</u>
<u>L5</u>	L4 and (multiuser or multi-user)	935	<u>L5</u>
<u>L4</u>	(database or data with base) near system	25169	<u>L4</u>
<u>L3</u>	L2 and transact\$ and session near manag\$	7	<u>L3</u>
<u>L2</u>	L1 and (user with profiles or user near profiles or history with profiles or history near profiles)	201	<u>L2</u>
<u>L1</u>	(multi-user or multiuser or shared or shar\$) near (data with base or database)	1986	<u>L1</u>

END OF SEARCH HISTORY